NCS TIB 96-1

NATIONAL COMMUNICATIONS SYSTEM

TECHNICAL INFORMATION BULLETIN 96-1

COLOR FACSIMILE

JANUARY 1996

OFFICE OF THE MANAGER
NATIONAL COMMUNICATIONS SYSTEM
701 SOUTH COURT HOUSE ROAD
ARLINGTON, VA 22204-2198

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE January 1996	3. REPORT TYPE AND DATES COVERED Final Report

4. TITLE AND SUBTITLE Color Facsimile	5. FUNDING NUMBERS DCA100-91-C-0031
6. AUTHOR(S) Stephen Perschau	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Delta Information Systems, Inc. 300 Welsh Road, Suite 120 Horsham, PA 19044-2273	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Communications System Office of Technology and Standards Division 701 South Court House Road Arlington, Virginia 22204-2198	10. SPONSORING/MONITORING AGENCY REPORT NUMBER NCS TIB 96-1
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The purpose of this project was to continue the work on color facsimile that was started in 1994. This report describes the creation and evaluation of default Huffman codes, which were completed in the last year. The development of default Huffman codes was a continuation of work performed in 1994. The modified Joint Photographic Experts Group (JPEG) software developed last year was used to compress images from two different classes. Newly developed software was then used on the resulting histograms to create composite Huffman code tables for the two images classes which could be used as defaults. This report is comprised of two section: (1) provides a brief description of the objectives of the task and an outline of the contents of this report; and, (2) describes the creation and performance of default Huffman coding tables for transmitting color FAX images by the JPEG (Joint Photographic Experts Group) baseline standard.

14. SUBJECT TERMS Facsimile Color Facsimile Color Fax Images	Test Images	15. NUMBER OF PAGES 60
		16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASS	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASS	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASS	20. LIMITATION OF ABSTRACT UNLIMITED

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet *optical scanning requirements*.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	PR	- Project
G	- Grant	TA	- Task
PE	- Program Element	WU	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD	- See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	- See authorities.
NASA	- See Handbook NHB 2200.2.
NTIS	- Leave blank.

Block 12b. Distribution Code.

DOD	- Leave blank.
DOE	- Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA	- Leave blank.
NTIS	- Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

NCS TECHNICAL INFORMATION BULLETIN 96-1

COLOR FACSIMILE

JANUARY 1996

PROJECT OFFICER

Stephen Perschau

STEPHEN PERSCHAU
Computer Scientist
Technology and
Standards Division

APPROVED FOR PUBLICATION:

Dennis Bodson

DENNIS BODSON
Chief
Technology and
Standards Division

FOREWORD

Among the responsibilities assigned to the Office of the Manager, National Communications System, is the management of the Federal Telecommunication Standards Program. Under this program, the NCS, with the assistance of the Federal Telecommunication Standards Committee identifies, develops, and coordinates proposed Federal Standards which either contribute to the interoperability of functionally similar Federal telecommunication systems or to the achievement of a compatible and efficient interface between computer and telecommunication systems. In developing and coordinating these standards, a considerable amount of effort is expended in initiating and pursuing joint standards development efforts with appropriate technical committees of the International Organization for Standardization, and the International Telegraph and Telephone Consultative Committee of the International Telecommunication Union. This Technical Information Bulletin presents an overview of an effort which is contributing to the development of compatible Federal, national, and international standards in the area of facsimile. It has been prepared to inform interested Federal activities of the progress of these efforts. Any comments, inputs or statements of requirements which could assist in the advancement of this work are welcome and should be addressed to:

Office of the Manager
National Communications System
Attn: N6
701 S. Court House Road
Arlington, VA 22204-2198

TASK 1
TECHNICAL WORK IN THE AREA OF FACSIMILE

SUBTASK 3
COLOR FACSIMILE

FINAL REPORT
CONTRACT DCA100-91-C-0031
OPTION YEAR 4

Submitted to:
NATIONAL COMMUNICATIONS SYSTEM
ARLINGTON, VA

January 1996

DELTA INFORMATION SYSTEMS, INC.
300 Welsh Road, Bldg. 3, Ste. 120
Horsham, PA 19044-2273
TEL: (215) 657-5270 FAX: (215) 657-5273

97-S-1154

TABLE OF CONTENTS

1 INTRODUCTION

This document summarizes work performed by Delta Information Systems, Inc. (DIS) for the National Communications System (NCS), Office of Technology and Standards. This office is responsible for the management of the Federal Telecommunication Standards Program, which develops telecommunication standards, whose use is mandatory for all Federal departments and agencies. The purpose of this project, performed under Task 2, Subtask 3 of contract number DCA100-91-C-0031 during Option Year 4, was to continue the work on color facsimile that was started in 1994.

This report describes the creation and evaluation of default Huffman codes, which were completed in the last year. The development of default Huffman codes was a continuation of work performed in 1994. The modified Joint Photographic Experts Group (JPEG) software developed last year was used to compress images from two different classes. Newly developed software was then used on the resulting histograms to create composite Huffman code tables for the two image classes which could be used as defaults.

This report is comprised of two sections.

Section 1.0 provides a brief description of the objectives of the task and an outline of the contents of this report.

Section 2.0 describes the creation and performance of default Huffman coding tables for transmitting color FAX images by the JPEG (Joint Photographic Experts Group) baseline standard.

2 DEFAULT HUFFMAN CODES

2.1 Background

In the 1994 study[1] eight test images were compressed with three candidates for default Huffman code sets. A "code set" is a set of four codes, one each for DC luminance, AC luminance, DC chrominance and AC chrominance. The three candidate Huffman code sets were:

-	T.81	JPEG default codes
-	Contribution D10	ITU-T Delayed Contribution from Japan
-	Delta Composite	

The Delta composite code set was derived from the combined histograms of the eight test images.

For any given test image, there was significant variation in the bit counts of the compressed data streams across the different Huffman code sets. That is, for a *given image*, the three Huffman code sets performed significantly differently. Typical differences were a few percent for images with data compression scale factors of 9 and 24 (low to medium compression, excellent to good image quality), and sometimes by more than 10 percent for a data compression scale factor of 71 (high compression, poorer image quality).

The theoretically optimal default Huffman code set is based on the total probability function of the symbols in a given symbol set. This probability function is the set of relative frequencies of the symbols over the "universe" of all images processed with all image processing parameters. A method of estimating such a probability function is presented later in this section. Had the candidate Huffman codes evaluated in 1994 closely approached the theoretical optimal code set, their performances for any given image and set of processing parameters would have been nearly identical.

Because of the significant differences in the performance of the test code sets, it was concluded that the three candidate code sets suffered from either or both of the following deficiencies: (1) they were derived from too few symbol samples, or (2) there was an insufficient mix of image characteristics and processing parameters that possess their own peculiar statistics.

2.2 Scope of Renewed Study

The 1995 study included: (1) obtaining and verifying an estimate of required sample size, (2) a measurement of the effects of image characteristics, and (3) the development of a theoretically sound procedure for creating a universal default Huffman code set that closely approaches the theoretical optimum.

The study employed some software tools developed in 1994 and 1995. These tools are summarized in Appendix A.

2.3 Image Model

In the present study, an image is assumed to have four symbol sets ("components"): DC luminance, AC luminance, DC chrominance and AC chrominance. In an actual CIELAB image, the "chrominance" comprises the A and B color components. The DC A and the DC B symbols are encoded by a common "DC chrominance" Huffman code; similarly, the AC A and the AC B symbols share a common "AC chrominance" code.

The JPEG compression and decompression software, as modified during the 1994 study, does not generate separate DC and AC histograms for the A and B components. Instead, as a byproduct of constructing the DC and AC chrominance Huffman codes, it produces DC and AC chrominance histograms, which are composites (sums) of the DC A and B, and the AC A and B, histograms respectively.

Because the 1995 study focuses on the number of bits generated just by Huffman coding, exclusive of other bits included in an actual compressed data stream, a "test image" in this study is not an image at all. Instead, the "image" is *represented* by the histograms produced when the actual image is compressed. The number of Huffman coded bits, for a given symbol set, is the sum of the products, over all symbols, of the number of times each symbol occurs, multiplied by the Huffman code word length for that symbol.

Since the modified JPEG software generates histograms for DC and AC "chrominance," and not separate A and B color components, the image model employed in this study lumps the A and B components (DC and AC) into DC and AC "chrominance" components. This lumping should not diminish the validity of the results and conclusions of this study, which evaluates the effects of random sampling and image characteristics on Huffman codes.

2.4 Sample Size

Determining an appropriate sample size consists of estimating the required number of symbols, selected at random from a much larger pool of symbols, to be statistically representative of the entire pool. This is different from selecting images from a large pool of images, because the image characteristics of the smaller set of images may be statistically significantly different from those of the large pool. To establish this statistical significance, one must estimate the random error associated with sampling symbols from a pool whose image characteristics are fixed. If this random error is small compared to differences arising from image characteristics, then the latter are statistically significant.

2.4.1 Theory

An analytical method of estimating the appropriate sample size is presented in Appendix B. This method yields an optimistic estimate of only a few thousand symbols, and a conservative estimate of high tens to high hundreds of thousands. This is small compared to the millions of symbol occurrences for each of the four symbol sets in the Delta composite histograms.

2.4.2 Random Sampling Results

A C program, SampHist.c, randomly samples, with replacement, symbols whose statistics are controlled by an input histogram set. The user specifies a percentage sampling rate that sets the number of samples to be taken for each image component. The program output is a set of histograms based on the sample results. If a possible symbol never occurs in the sample, it is assigned a count of 1.

SampHist.c simulates the following experiment: Mix a very large number of marbles labeled with symbol names in a box, with the proportion of marbles with a given symbol name specified by the input histogram. Draw N times from the box, each time noting the symbol name, and then throwing the marble back in the box and remixing the marbles before drawing again. Build a new histogram based on the results of the drawings.

This program was employed to randomly sample a symbol pool at sampling rates of 50 percent and 10 percent, with the symbol pool statistics determined by the Delta composite histograms. Huffman code tables (consisting of code word lengths, not the codes themselves) for each image component were generated from the original composite histograms, the 50 percent histograms, and the 10 percent histograms. AvHuflen.c, a C program written in 1994, was then run to compute the average number of Huffman coded bits per symbol when the "image" was always represented by the whole composite histogram set (one histogram for each of the four image components), and the Huffman codes were the three sets mentioned above. The results for the original composite and the 50 percent sample were identical in bit count for all components. The 10 percent sample results agreed exactly for the DC luminance, differed by 1 bit out of approximately 3 million for DC chrominance, by approximately 2400 bits out of 19.9 million, or 0.012 percent for the AC luminance, and roughly 2000 bits out of 11.7 million, i.e., 0.017 percent for the AC chrominance. These differences were much smaller than those observed in 1994 when different Huffman code sets encoded a given image. The results show that performance differences among the code sets tested in 1994 were almost certainly due to differing image characteristics and/or processing parameters in the image sets used to derive these codes.

2.4.3 JPEG Huffman Coding Anomaly

In the random sampling experiment described above, a startling result occurred. The Huffman codes derived from 10 percent sampling of the Delta composite performed slightly *better* than those derived from the full histogram when the "test image" was represented by the full histogram itself. This anomaly occurred in the AC luminance and AC chrominance components. Even though the differences were very small, the fact that another Huffman code could do even very slightly better than the full histogram's own Huffman code caused considerable consternation.

A copy of AvHuflen.c was modified to derive pure Huffman code word lengths from a supplied histogram, instead of reading the JPEG Huffman code word lengths from a file. Pure Huffman codes are not constrained to the JPEG length limit of 16 bits. When the code performance comparisons were repeated with pure Huffman codes, the anomaly disappeared.

Spreadsheets were created to compare the bit counts produced by various Huffman codes for each symbol of the AC luminance symbol set, and to show the total differences. The comparisons were:

Pure Huffman, full histogram v. pure Huffman, 10 percent sample,
JPEG v. pure Huffman, both derived from the full histogram,
JPEG v. pure Huffman, both derived from the 10 percent sample,
JPEG Huffman, full histogram v. JPEG Huffman, 10 percent sample.

In all four cases, the encoded "image" is represented by the full, unsampled composite histogram, i.e., the Delta composite from the 1994 project.

The first comparison showed that the pure Huffman code behaved as expected, namely, that the code from the 10 percent sample performed very slightly worse than that from the full histogram when encoding an "image" whose statistics are represented by the latter histogram.

The second and third comparison showed that the JPEG Huffman code does worse than the pure code when both codes are derived from either the full or the 10 percent histogram. The degradation was worse, however, for the full histogram than for the 10 percent sample, enough worse to cause the anomaly.

In the fourth spreadsheet, there were only four symbols in which the code word lengths differed. Table 2.1 shows these differences.

TABLE 2.1 - SYMBOLS IN WHICH THE CODE LENGTH DIFFERS

Symbol No. (hex)	Number of occurrences	Code word length (full histogram)	Code word length (10 percent sampled)	Bit count Difference
25	486	16	14	972
34	753	15	16	-753
E1	1478	15	12	4434
F0	2252	11	12	-2252

The Huffman code word lengths derived from the full histogram produced 2401 more bits than did those derived from the 10 percent sample.

It is evident, therefore, that the JPEG Huffman coding algorithm, with its 16 bit limit on the code word length, may produce sub-optimal codes in the sense that a code produced from a given "image" (e.g., the full composite histogram) may not perform quite as well when encoding *that same* image as code produced by another "image" with nearly identical statistics (e.g., the 10 percent sampled histogram). However, the difference is tiny compared with the various differences observed in the 1994 Color Facsimile project.

2.5 The Effects of Image Characteristics

Busy and bland images aptly illustrate how image characteristics affect JPEG symbol statistics. Busy images exhibit rapid spatial variations in brightness and/or hue; bland images are the opposite. These two image classes are easily distinguished by visual inspection. Of course, there are intermediate images, parts of which are busy and parts bland, and there are degrees of busyness. The images evaluated for the 1995 study were deliberately chosen to be obviously busy or obviously bland.

Busy images are inherently harder to compress than bland images, because, as is explained presently, the former generate more symbols to be encoded than the latter. A universal default Huffman code should take this into account by favoring busy image statistics according to the proportion of busy image symbols. Section 2.6 shows how to determine the extent to which busy image statistics should be favored.

A DC symbol represents the difference between the quantized DC coefficient of the block being encoded and that of the block that was most recently encoded. Because the DC coefficients vary more from block to block in busy images than in bland, the probability of the zero-value symbol (SSSS = 0) is considerably less in busy than in bland images.

For AC, busy images produce more non-zero quantized coefficients than do bland images. Consequently, busy images produce more AC symbols for a given image size, since, in any given block, a symbol is generated for each non-zero coefficient.

Each block almost always includes an end-of-block symbol in both busy and bland images. (In the very rare case of the last coefficient in a block being non-zero, an end-of-block symbol is not encoded.) Since busy images generate, on the average, more AC symbols per block, the probability of the end-of-block symbol is significantly less in busy than in bland images.

Evaluation of the statistical differences between busy and bland images commenced with the selection of a set of 7 obviously busy and 7 obviously bland images, all fully sampled. Table 2.2 describes the characteristics of each of the 14 color images chosen. All images were first converted to the CIELAB color space. The DLB extension on the filenames was added to indicate that the images are in the RAW format with the Delta header. Hard copies of the images (in grayscale) are included in Appendix C of this report.

The images were compressed with the modified JPEG software at a compression scale factor of 25 (the JPEG recommendation for satisfactory image quality and good compression) to build a histogram for each symbol set of each image. Program CHistv2 then produced composite histograms for the busy image set and the bland image set. In the composite histograms, possible symbols that never occurred were assigned counts of 1 to guarantee a Huffman code for each possible symbol.

TABLE 2.2 - BUSY AND BLAND IMAGE SET DESCRIPTION

File Name	Description	Image Type	Image Size
img0009.dlb	boats	busy	1504x2048
img0005.dlb	fish	busy	1504x2048
img0024.dlb	train in warehouse	busy	1504x2048
img0012.dlb	handbags	busy	1496x2040
img0021.dlb	violin & tapestry	busy	1504x2032
n8.dlb	woman in photo	busy	1504x2048
bikerace.dlb	bike race	busy	1504x2048
img0008.dlb	water	bland	1520x2048
img0003.dlb	floppy disk	bland	1520x2048
img0020.dlb	2 violins	bland	1520x2048
n6.dlb	flower	bland	1520x2048
sunset.dlb	sunset	bland	1520x2048
plane.dlb	airplane	bland	2560x1216
gcanyon.dlb	grand canyon	bland	1520x2048

The next step was to run SampHist.c to perform random sampling at 50 and 10 percent for both image classes. For each class, separate Huffman code sets were generated from the original composite histogram set and the 50 and 10 percent samples, and AvHuflen.c measured the performances of the three code sets against the original composite histogram set. The Huffman codes derived from the 50 percent samples gave results, for both the busy and bland composites, that agreed in average bits per symbol with the results of using the Huffman codes from the original histograms to within 0.001 bit for all symbol sets. The results of using the Huffman codes derived from the 10 percent samples agreed to within 0.001 bit per symbol in all symbol sets, except, in the bland composite, the AC luminance and AC chrominance results differed by 0.002 bit. Therefore, the "noise level" due to random sampling is no more than 0.002 bit per symbol.

In the final step, the Huffman code set generated by the full busy composite was designated "busy Huffman code," and that produced by the full bland composite was called "bland Huffman code." The Huffman code sets from the 50 and 10 percent histograms were discarded. AvHuflen.c was then run with both the busy and bland Huffman codes, with test "images" represented by the following histogram sets: (1) the busy composite, (2) each busy image, (3) the bland composite and (4) each bland image.

The initial plan specified that all images have the same number of pixels, and that both the width and height of each image be a multiple of 8 pixels. The second specification ensures that each image component produces a whole number of 8 x 8 blocks without the JPEG compression software's having to fill out partial blocks.

The goal of all the images having the same number of pixels was not met. All bland images produced 48128 blocks per component, as did five of the seven busy images. The remaining two busy images generated 47685 and 47752 blocks per component. The goal of each image dimension being a multiple of 8 was met.

Because of the differing sizes, the data for the number of symbols are expressed in average symbols per block instead of total number of symbols. In the following tables, the number of luminance blocks in an image is the total number of pixels divided by 64. The number of chrominance blocks is twice this value, because the JPEG program generates separate blocks for the A and B color components, i.e., two "chrominance" blocks for each luminance block. The average number of symbols per block is always 1 for the DC luminance and the DC chrominance, because exactly one DC symbol (the difference between the quantized DC coefficients in the current and previous block) is encoded per block.

In the busy (bland) composite, the numbers of luminance and chrominance blocks are the sums of the corresponding numbers over all busy (bland) images. However, in the composites, each possible symbol that never occurs in any image comprising the composite is assigned a value of 1 for the number of occurrences. This slightly increases the total number of occurrences, hence very slightly increases the average number of symbols per block. This effect is so small, however, that the average number of symbols per DC block computes to 1.0 with several more zeros before another non-zero digit occurs, and hence is negligible.

Table 2.3 shows the results of encoding the busy composite and the individual busy images with both the busy and the bland Huffman codes. Table 2.4 is a similar table for the bland composite and images.

TABLE 2.3

Results of Encoding Busy Images with Busy and Bland Huffman Codes

Image (or composite)	Number of Blocks Luminance -------------- Chrominance	Symbol Set	Average Symbols per Block	Average bits per symbol (busy Huffman code)	Average bits per symbol (bland Huffman code)
Busy Composite	336077	DC Luminance	1	2.981	3.690
	----------	AC Luminance	12.71	3.374	3.695
	672154	DC Chrominance	1	2.344	2.490
		AC Chrominance	3.03	2.737	2.822
Boats	48128	DC Luminance	1	2.982	3.483
	--------	AC Luminance	13.63	3.384	3.655
	96256	DC Chrominance	1	2.202	2.203
		AC Chrominance	2.76	2.699	2.716
Fish	48128	DC Luminance	1	2.936	3.550
	--------	AC Luminance	9.48	3.331	3.462
	96256	DC Chrominance	1	2.318	2.474
		AC Chrominance	2.75	2.630	2.630
Train in Warehouse	48128	DC Luminance	1	3.041	3.260
	--------	AC Luminance	9.14	3.442	3.568
	96256	DC Chrominance	1	2.225	2.160
		AC Chrominance	2.42	2.672	2.618
Handbags	47685	DC Luminance	1	2.829	3.719
	--------	AC Luminance	12.41	3.318	3.676
	95370	DC Chrominance	1	2.295	2.401
		AC Chrominance	3.16	2.723	2.843
Violin & Tapestry	47752	DC Luminance	1	3.019	3.978
	--------	AC Luminance	11.35	3.338	3.670
	95504	DC Chrominance	1	2.330	2.561
		AC Chrominance	2.97	2.633	2.662
Woman in Photo	48128	DC Luminance	1	2.983	3.970
	--------	AC Luminance	25.44	3.406	3.919
	96256	DC Chrominance	1	2.426	2.605
		AC Chrominance	3.55	2.948	3.126
Bike Race	48128	DC Luminance	1	3.074	3.876
	--------	AC Luminance	7.51	3.360	3.526
	96256	DC Chrominance	1	2.613	3.023
		AC Chrominance	3.63	2.779	2.996

TABLE 2.4

Results of Encoding Bland Images with Busy and Bland Huffman Codes

Image (or composite)	Number of Blocks Luminance / Chrominance	Symbol Set	Average Symbols per Block	Average bits per symbol (busy Huffman code)	Average bits per symbol (bland Huffman code)
Bland Composite	340480	DC Luminance	1	3.256	2.414
		AC Luminance	6.23	3.452	3.217
	680960	DC Chrominance	1	2.095	1.815
		AC Chrominance	2.28	2.564	2.414
Water	48640	DC Luminance	1	3.451	2.036
		AC Luminance	4.43	3.950	3.405
	97280	DC Chrominance	1	2.000	1.277
		AC Chrominance	1.16	2.138	1.354
Floppy Disk	48640	DC Luminance	1	3.311	2.464
		AC Luminance	6.22	3.306	3.177
	97280	DC Chrominance	1	2.202	1.813
		AC Chrominance	2.06	2.587	2.452
Two Violins	48640	DC Luminance	1	3.099	2.670
		AC Luminance	6.92	3.437	3.271
	97280	DC Chrominance	1	2.192	2.139
		AC Chrominance	2.24	2.526	2.390
Flower	48640	DC Luminance	1	3.113	2.698
		AC Luminance	9.15	3.557	3.416
	97280	DC Chrominance	1	2.094	1.681
		AC Chrominance	1.54	2.334	1.876
Sunset	48640	DC Luminance	1	3.417	2.210
		AC Luminance	4.71	3.405	3.037
	97280	DC Chrominance	1	2.043	1.772
		AC Chrominance	2.55	2.623	2.520
Airplane	48640	DC Luminance	1	3.278	2.412
		AC Luminance	5.29	3.337	3.066
	97280	DC Chrominance	1	2.084	1.973
		AC Chrominance	2.96	2.646	2.632
Grand Canyon	48640	DC Luminance	1	3.119	2.405
		AC Luminance	6.88	3.256	3.047
	97280	DC Chrominance	1	2.051	2.048
		AC Chrominance	3.42	2.703	2.733

The composite data serve as an indication of the average performance of busy and bland images compressed with busy and bland Huffman codes. The most striking difference was in the DC luminance: the degradation suffered from encoding the busy composite with the bland Huffman code or vice versa was in the order of 30 percent. The other symbol sets gave considerably less degradation, in the neighborhood of 5 to 10 percent.

The individual image data produced some surprises. Among the busy images, the Train image DC and AC chrominance did slightly better with the bland than with the busy Huffman codes. The AC chrominance for the Fish image scored a dead heat with the two Huffman codes. The DC chrominance results in the Boats image differed by 0.001 bit, which is not statistically significant because of the random sampling uncertainty. In the bland set, the AC chrominance of the Grand Canyon image produced 0.01 fewer bits per symbol with the busy Huffman code than with the bland. These small "reversals" in the chrominance data may have occurred because the luminance data probably contribute more than the chrominance to a visual judgement of whether an image is busy or bland.

2.6 Designing a Universal Huffman Code Set

Building a universal default Huffman code set (one code for each image component) to cover all image classes and processing parameters is straightforward in principle when one employs a probability tree to obtain the *total probability* of each symbol in a symbol set.[2] The method is illustrated by an example.

Let all images be classified into two or more mutually exclusive classes (no image can belong to more than one class). The current example assumes two classes, c1 and c2, for example, busy and not busy. Assume that the processing parameters are characterized by three compression scale factors and two sub-sampling rates. Let f1, f2 and f3, for example 8, 25 and 71, be the compression scale factors; let r1 and r2, e.g., 1:1:1 and 4:2:2, be the sub-sampling rates.

Each combination of class, scale factor and sub-sampling rate may produce significantly different statistics for the encoded symbols. Consequently, a separate default Huffman code set for each combination would be expected to do better when used to encode images characterized by that combination than one "universal" code set. However, if a single default code set is required, the procedure described below leads to the best compromise across all image characteristics and processing parameters.

2.6.1 Required Data

The required data are:

- Conditional probability estimates
- Composite histograms

The present example assumes that the image sub-sampling rate is conditioned on the compression scale factor, which, in turn, is conditioned on image class. This is a purely arbitrary assumption for illustrative purposes only. If other dependencies become evident, then the probability tree must be restructured accordingly.

For this example, the required probability estimates for each image component are:

c1 and c2 (unconditional),
f1, f2 and f3 given c1, and f1, f2 and f3 given c2,
r1 and r2 given c1 and f1, r1 and r2 given c1 and f2,
. . .
r1 and r2 given c2 and f3.

In all, there are 20 probabilities to be estimated: 2 for the classes, 6 for the compression scale factors (3 for each class) and 12 for the sub-sampling rates (2 for each compression scale factor and each class). For simplicity, one might assume that the various parameters are statistically independent, in which case there are only 3 scale factor and 2 sub-sampling rate probabilities. The example given here is not restricted to this assumption.

A composite histogram for each image component is compiled from several images for each of the 12 combinations of class, scale factor and sub-sampling rate. From each such histogram an estimated symbol probability function is computed.

2.6.2 The Probability Tree

Figure 2.1 shows the probability tree. At the bottom of the tree there is a "leaf" for every symbol in the set for each of the 12 combinations; for clarity, the figure shows a leaf representing just one symbol for each combination.

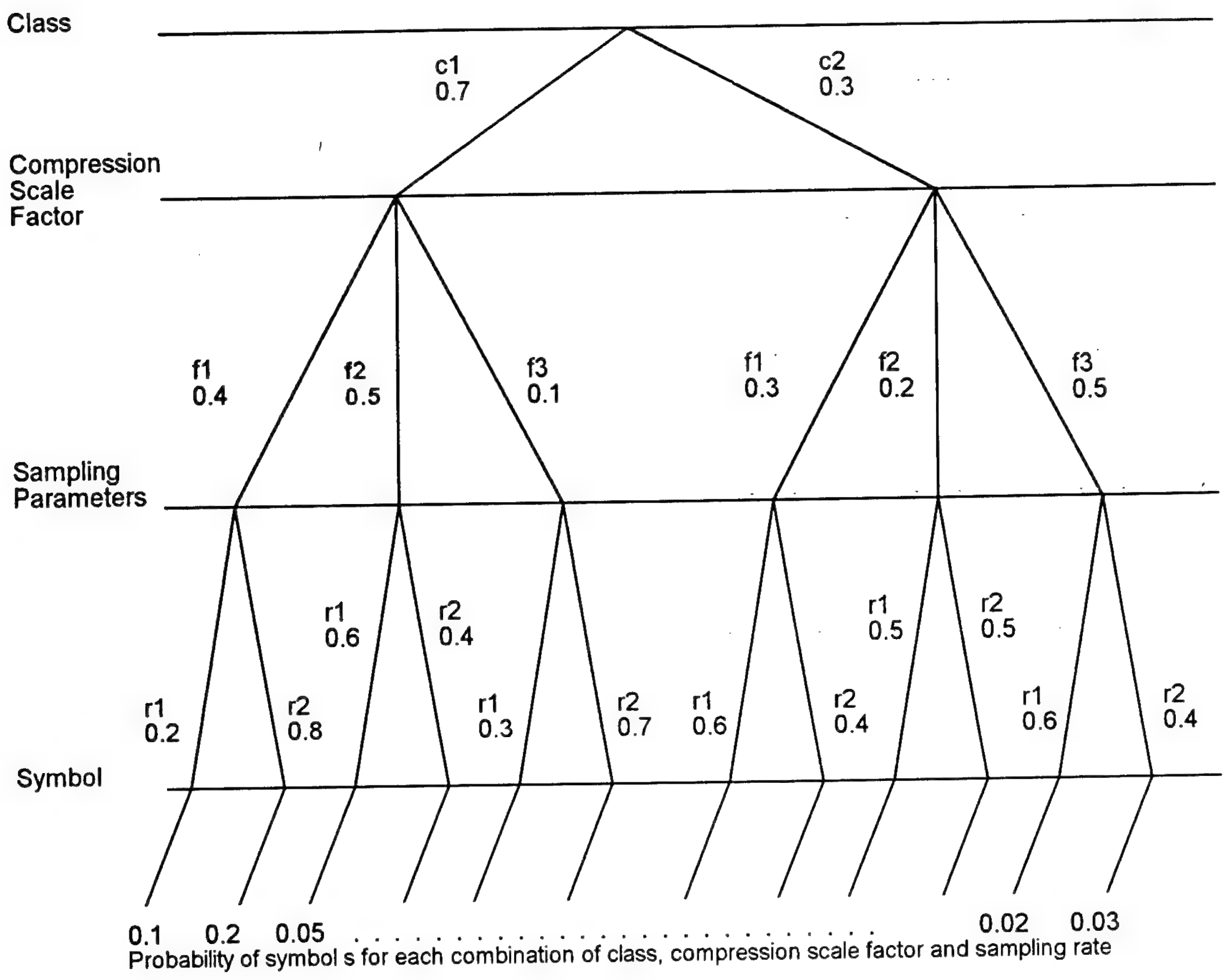


Figure 2.1 Probability Tree for Image Classes and Processing Parameters

The general rules for any probability tree are:

- The probability associated with any tree branch is the probability of the event represented by that branch, *given* the set of events associated with the node from which the branch leads.
- The sum of the probabilities associated with all branches leading away from any one node is always 1.
- The probability of passing through the root (top) node is 1.
- The probability of passing through any other node is the product of the usually conditional probabilities associated with all the branches in the path from the root to the node. (The probabilities associated with branches

leading from the root node are unconditional, since the probability of passing through the root node is 1.)

- Events represented by a tree node may be conditioned on events represented by higher-level nodes in the tree (toward the root node), but not by lower-level nodes.

To compute the total probability of symbol s, one forms a sum of products. Each product is the probability of the symbol given one of the (in this example) 12 combinations of class, scale factor and sub-sampling rate, times the probabilities associated with the path leading from that combination back to the tree root.

In the example shown in the figure, the various probabilities are chosen arbitrarily for illustrative purposes only. With some terms missing, the method of computing the total probability for one symbol is shown:

$$\begin{aligned} tp(s) &= (0.1 \times 0.2 \times 0.4 \times 0.7) \\ &+ (0.2 \times 0.8 \times 0.4 \times 0.7) \\ &+ (0.05 \times 0.6 \times 0.5 \times 0.7) \\ &+ 7 \text{ terms} \\ &+ (0.02 \times 0.6 \times 0.5 \times 0.3) \\ &+ (0.03 \times 0.4 \times 0.5 \times 0.3). \end{aligned}$$

With accurate symbol probability estimates for the 12 combinations of image class and processing parameters (from the 12 composite histograms), and good conditional probability estimates (f1 given c1, r2 given f2 given c2, etc.), one can compute the total probabilities of all the symbols, and from this probability function, derive an optimal Huffman code that is universal over all image class and processing combinations.

2.6.3 Estimating Tree Branch Probabilities

When estimating the probabilities for the probability tree branches, one must remember that one is in effect counting *symbols*, not images. For example, suppose that the two image classes, c1 and c2, represent busy and not-busy images. The probability of c1 is *not* defined as the probability that an *image* is busy, but rather as the proportion of *symbols* that are generated by busy images. Busy images tend to generate more AC symbols than do bland images, because the former produce more non-zero AC spatial frequency coefficients.

For the DC components, the number of symbols is independent of whether the image is busy or bland; therefore, c1 and c2 *are* the probabilities that the image is busy and bland respectively.

The following treatment shows how to estimate the c1 and c2 probabilities for AC luminance or chrominance in terms of the probability that an image is busy, while allowing for the difference in symbol production by busy and bland images.

Assume that the following quantities are known, for AC luminance or AC chrominance:

p(bi)	probability that an image is busy,
p(ni)	probability that an image is bland (not busy),
n(b)	average number of symbols produced by a busy image,
n(n)	average number of symbols produced by a bland image,

where, for n(b) and n(n), the images are assumed to be of the same size. If this assumption is false, then size related weighting factors must be included. Absolute values of n(b) and n(n) are not required; only the following ratios:

R(b) = n(b) / [n(b) + n(n)],
R(n) = n(n) / [n(b) + n(n)].

It is now shown that the symbol probabilities for c1 and c2 (busy and bland) are given by:

p(busy symbol) = p(bi)R(b) / [(p(bi)R(b) + p(ni)R(n)],
p(bland symbol) = p(ni)R(n) / [(p(bi)R(b) + p(ni)R(n)].

Assume that, in the following imaginary experiment, I(b) busy images and I(n) bland images are processed. Let n(b) be the average number of symbols per busy image, and n(n) be the average number per bland image. Let p(bi) and p(ni) be estimated by the ratios

p(bi) = I(b) / [I(b) + I(n)]

and

p(ni) = I(n) / [I(b) + I(n)].

Then:

Total number of busy image symbols = n(b)I(b),
Total number of bland image symbols = n(n)I(n),
Total number of all image symbols = n(b)I(b) + n(n)I(n).

To estimate the probabilities of busy and bland image symbols from this experiment, one would compute the following ratios:

p(busy symbol) = total busy image symbols / total symbols, p(bland symbol) = total bland image symbols / total symbols,

from which

p(busy symbol) = n(b)I(b) / [n(b)I(b) + n(n)I(n)],
p(bland symbol) = n(n)I(n) / [n(b)I(b) + n(n)I(n)].

Dividing the numerators and denominators of both expressions by the product

[(n(b) + n(n)][I(b) + I(n)]

gives

p(busy symbol) = p(bi)R(b) / [(p(bi)R(b) + p(ni)R(n)],
p(bland symbol) = p(ni)R(n) / [(p(bi)R(b) + p(ni)R(n)].

Estimating the probability that a symbol comes from an image compressed by a given scale factor requires similar treatment, because low scale factors produce more AC symbols than do high scale factors. Similarly, different sub-sampling rates produce different numbers of symbols.

2.7 Default Huffman Coding Conclusions

An optimal universal default Huffman code for any given symbol set is that which comes from the total probability function of the symbols. Unlike many random processes, like games of chance, drawing names from a hat, or radioactive decay, the probability function of a JPEG symbol set does not lend itself to theoretical derivation. There is no *a priori* knowledge of the proportions in which various kinds of images and image processing parameters will occur. Estimating symbol probabilities even for one image by analytical means, without actually counting symbols, is probably not feasible. Consequently, empirical, rather than analytical, means are required to estimate the total probability function.

A brute force method of estimating this probability function would be to collect histograms for all the images processed by all sites over a long period of time and compile a composite histogram. Such an estimate would ensure a mix of image characteristics and processing parameters that would be representative of the "universe." Unfortunately, such a scheme is economically impractical, because every facsimile machine would have to be equipped with the extra hardware and software required to collect the histograms. Selected sites could be so equipped; however, this might introduce site-dependent statistics. There would have to be enough sites to average out this effect. Each such site would have to save its histograms and send them to a processing center for compilation after the test period expires.

The probability tree approach offers the following advantages over the brute force method:

- A few selected test sites would gather the composite histograms for the various combinations of image characteristics and processing parameters. The random sampling theory and experiments reported above show that only a few images would be required for each combination.

- If the image classes are readily observable by visual inspection (for example busy/bland), a large number of sites could, for each of hundreds, possibly thousands, of images processed, log the image classes and processing parameters, and send the logs to a processing center. Such logs could be entered separately from the facsimile machines, for example, into text files on small computers. The processing center could then design the probability tree.

Even the probability tree approach is not simple, because it may be difficult to estimate all the conditional probabilities. Various simplifying assumptions may be required; such assumptions would, unless realistic, compromise the universality of the default codes.

In the 1994 project, all three tested code sets performed, for a given image, mostly within a few percent of each other, and always within approximately ten percent of one another. If such variations are acceptable, then an adequate default code set can be derived by any facility by compiling composite histograms from tens to hundreds of images, representing a thorough mix of image characteristics and processing parameters. To combat the "not invented here" syndrome, a compromise code set could be derived by combining the composite histograms of all contributors, all sharing credit for the resulting Huffman codes.

If further effort toward better performance is to be undertaken, then the probability tree approach is recommended. Several contributors should independently develop Huffman codes, and all should be tested with each of many images. If, over the various test images, the performances of all codes with any given image are nearly the same (for example to within a few tenths of a percent), then these codes closely approach the optimum, and, as suggested above, a joint code can be developed, with the contributors jointly claiming credit.

To avoid bias toward any contributor, the images used to test the codes should be chosen at random from a set that excludes all images from which the candidate codes are derived.

REFERENCES

[1] National Communications System, *Color Facsimile*, NCS TIB 95-2, February 1995.

[2] C. Ash, *The Probability Tutoring Book*, IEEE Press, pp.56-58.

APPENDIX A

SOFTWARE TOOLS

This appendix summarizes the various computer programs employed during this study.

JPEG Software Modifications

In 1994 the JPEG compression and decompression software was modified for the following purposes: (1) process or generate the Delta Information Systems CIELAB image file format as discussed in Section 3.1.1 of the 1994 final report; (2) save into a text file a histogram of symbol occurrences used to generate each Huffman code table; (3) use such a histogram text file to generate Huffman tables, and (4) to save Huffman tables to a text file.

CHistv2.c

This C program, written in 1994, reads a list of file names of histogram sets (one histogram for each Huffman code table) and builds a composite histogram set by, for each symbol in each symbol set, summing the number of occurrences in all the files named in the list. After the summing is completed, the program checks each *possible* symbol, and if it has a count of 0, it sets it to 1. This ensures that, when the composite histogram set produces Huffman codes, a code word is generated for every possible symbol.

SampHist.c

This program, developed in 1995, is described in the body of this report. The user specifies a histogram file and a number representing the sampling percentage, for example 10 for 10 percent. The program determines, for each histogram, the number of samples required to be the specified percentage of the total symbol count in that histogram.

Because the sampling process is with replacement (a "marble" is "thrown back into the box" after being "drawn"), specifying 100 percent does not guarantee that the output histogram will be identical to the input. That is why this report refers to the "whole" or "unsampled" histogram, or similar words, and never "100 percent sampling."

AvHuflen.c

Written in 1994, this C program measures the performance of a Huffman code. The total bit count produced by the JPEG image compression and decompression program includes, among other things, SSSS bits per non-zero quantized coefficient, where SSSS is the "size" of the coefficient. Moreover, the bit count combines the bits generated by all image components. AvHuflen.c, on the other hand, counts the bits produced just by the Huffman coding process and reports the results separately for each image component. The only drawback of employing AvHuflen.c is the lumping of the A and B DC and AC color components of an actual image into DC and AC "chrominance" as explained earlier.

The program reads a Huffman code file produced by the modified JPEG program and constructs, for each symbol set, a one-dimensional array showing the code word length for each symbol (0 for impossible symbols). The program then reads a text file containing a list of image histogram file names. For each such histogram file, and for each symbol set, AvHuflen.c sets the total number of bits equal to the sum of the products, over all symbols, of the number of occurrences of each symbol times the code word length for that symbol. The program reports, for each component of each image, the total number of encoded bits, the total number of encoded symbols, and the average number of bits per symbol.

APPENDIX B

ESTIMATING SAMPLE SIZE

1 Introduction

The following theoretical analysis gives an estimate of the required number of symbols, randomly sampled with replacement from a large pool, to adequately represent the pool. "Adequately represent" means to produce a Huffman code that performs nearly as well as a Huffman code derived from the entire pool when both codes encode symbols whose statistics are identical to those of the pool.

The purpose of this analysis is to ensure that the sample size is large enough to make random errors due to sample size small compared to differences arising from different image characteristics or processing parameters. Sampling with replacement ensures that the statistics of the pool being sampled are independent of samples already selected.

2 The Estimation Method

To simplify the analysis, a coding model based on the information content of each symbol is assumed instead of Huffman coding. The information content of symbol s is:

$$I(s) = -\log_2 p(s),$$

where p(s) is the probability of the symbol. The coding model has code "words" I(s) bits long. This is clearly not realizable in practice, at least in systems that employ a specific binary code for each symbol, because I(s) is not necessarily an integer.[*]

With information coding, the average number of bits per symbol is given by:

$$H = \text{Sum}\,[-p(s)\log_2 p(s)],$$

which is the entropy of the symbol set, and is the theoretical minimum average bit rate when the symbols are coded independently. It is well known that the optimal Huffman code always produces an average number of bits per symbol that is at most 1 greater than H.

Now, let the code word lengths I(s) be derived from an *estimated* probability

[*] Arithmetic coding, which in effect encodes the entire message as a long binary fraction, can very closely approach the theoretical minimum average number of bits per symbol.

function for the symbols. Let the estimated probability for a given symbol be given by

$$q(s) = p(s) + dp(s),$$

where dp(s) is the estimation error. Then, the average number of bits per symbol over the very long term is given by:

$$B = \text{Sum } \{-p(s) \log_2 [p(s) + dp(s)]\}.$$

We now approximate B under the assumption that $|dp(s)| < p(s)$. First, it is noted that

$$\log_2 x = \ln x / \ln 2 = K \ln x,$$

where ln x is the natural logarithm (base e) of x, and $K = 1.44269\ldots$.

The Taylor's series expansion of ln (p + dp) about ln p is:

$$\ln (p + dp) = \ln p + dp / p - dp^2 / 2p^2 + dp^3 / 3p^3 - dp^4 / 4p^4 + \ \ldots$$

valid for $|dp| < p$. Substituting into the equation for B yields:

$$\begin{aligned} B = K \text{ times } \{ & \\ & \text{Sum } [-p(s) \ln p(s)] \\ & - \text{Sum } [dp(s)] \\ & + \text{Sum } [dp(s)^2 / 2p(s)] \\ & - \text{Sum } [dp(s)^3 / 3p(s)^2] \\ & + \ldots\}. \end{aligned}$$

The first sum in the braces, multiplied by K (ratio of $\log_2 x$ to ln x), is H, the entropy. The second sum vanishes, because the sum of the probabilities in both the actual and estimated probability functions is 1; therefore, the sum of the estimation errors is 0.

The contribution of each symbol to B - H, the excess average bits per symbol over the entropy, is therefore given by b(s):

$$\begin{aligned} b(s) &= K [dp(s)^2 / 2p(s) - dp(s)^3 / 3p(s)^2 + dp(s)^4 / 4p(s)^3 - \ldots] \\ &= K [dp(s)^2 / 2p(s)] [1 - (2/3)x + (2/4)x^2 - (2/5) x^3 + \ldots] \end{aligned}$$

where x is defined as dp(s)/p(s), and Sum [b(s)] = B - H.

Consider the infinite series inside the right set of brackets:

$$1 - (2/3)x + (2/4)x^2 - (2/5)\,x^3 + \ldots\ .$$

For $x > 0$ (but less than 1, of course, for the Taylor's series to converge), the terms alternate in sign and decrease in magnitude as the power of x increases. The series sum is therefore bounded by

$$1 - (2/3)x < \text{sum} < 1 - (2/3)x + (2/4)x^2.$$

For $x < 0$, all the terms are positive. Because the coefficients decrease with increasing powers of x, the sum is less than the geometric series in $|x|$:

$$1 + |x| + |x|^2 + |x|^3 + \ \ldots \ = 1 / (1-|x|) \text{ for } |x| < 1.$$

Therefore, for $x < 0$ (and $|x| < 1$), the infinite series is bounded by

$$1 < \text{sum} < 1 / (1-|x|).$$

For $|dp(s)| / p(s)$ less than 0.1, the infinite series sum is, to good approximation, 1 to within an error of approximately $|dp(s)| / p(s)$. With $|dp(s)| / p(s) = ½$, the sum is bounded below by 2/3 and above by 2. Therefore, to within a roughly 2 to 1 range, the contribution of each symbol to the excess number of bits per symbol is given by:

$$b(s) = \text{(approx.) } K\, dp(s)^2 / 2p(s).$$

It is now shown that the ratio of the expected value of $dp(s)^2$ to $p(s)$ is almost independent of s. The number of times, n(s), symbol s occurs in a sample (with replacement) of size N has a binomial probability function, i.e., the probability of k successes in N trials, given the probability, p, of a success in one trial. In the current context, a trial consists of selecting and replacing any symbol at random, a success consists of finding that the selected symbol is symbol s, the number of successes is n(s), and the number of trials is the sample size. The mean and variance formulas given below are properties of the binomial function, but are couched in terms of the symbol statistics.

The expected value (mean) of n(s) in a sample of size N is:

$$E[n(s)] = p(s)N,$$

and the variance, which is the expected value of $[n(s) - p(s)N]^2$ is

$$E\{[n(s) - p(s)N]^2\} = p(s)[1 - p(s)]N.$$

Factoring out N gives

$$n(s) - p(s)N = N[\, n(s) / N - p(s)] = Ndp(s).$$

This is because n(s) / N is the estimated probability of symbol s based on counting the number of times symbol s occurs in a total of N symbols. Consequently,

$$E\{[n(s) - p(s)N]^2\} = E[N^2\, dp(s)^2] = p(s)[1 - p(s)]N,$$

from which

$$E[dp(s)^2] = p(s)[1 - p(s)] / N$$

and

$$E[dp(s)^2] / p(s) = [1 - p(s)] / N.$$

For $p(s) << 1$, $1 - p(s)$ is approximately 1; hence the expected contribution of symbol s to the excess bit rate is approximately 0.721 / N, and for p(s) nearer to 1, the contribution is less than 0.721 / N, where 0.721... is K / 2. Therefore, for dp(s) fairly small compared to p(s), the total number of excess bits is given, roughly, by:

$$B - H < 0.721 \text{ times Sum } (1 / N);$$

whence

$$B - H < 0.721\ S / N$$

where S is the number of symbols in the set, and N is the number of samples. For $S = 256$ (more than the maximum for JPEG AC coefficients), this analysis shows that only about two thousand samples are required to create a code with which $B - H < 0.1$; i.e., the estimated probability function would yield a code that is less than 0.1 bit per symbol worse than the entropy. This estimate is optimistic, however, as is shown below.

3 Discussion

The derivation presented above assumes that: (1) the Taylor's series expansion for ln (p + dp) converges, which requires that $|dp| < p$, and (2) that $|dp|$ is fairly small compared to p, so that the approximation for the excess bits is reasonably accurate. (If condition (2) is satisfied, then condition (1) is also.)

The *expected* number of occurrences of symbol s in N samples is Np(s). However, when p is small, and N is not sufficiently large, symbol s may not occur at all in the sample. The estimated probability of s would then be 0, dp would be minus p, the series would not converge for that symbol, and $|dp| / p$ would be 1, violating condition (2).

However, as p(s) approaches 0, p(s) log p(s) (any logarithm base) approaches 0 even though the logarithm approaches minus infinity. For example, if p(s) is 2^{-14}, the code word length is 14 bits, but the symbol occurs so rarely that it contributes only 14×2^{-14}, or approximately 0.0009 bit, to the total bit rate. Therefore, symbols having very small probabilities may be ignored in estimating the sample size.

The Poisson probability function is an approximation of the binomial function when $p << 1$. It expresses the probability of n(s) occurrences of symbol s in N samples in terms of $m = p(s)N$, the expected number of occurrences. (In printed literature, the lower-case Greek letter lambda is often used instead of m.) For p small, N large, and m intermediate, the Poisson function is a good approximation of the binomial function.

To assure, with high probability, that a symbol will occur at least once, and preferably several times in the sample, the value of m should be 10 or greater. Table B.1 was derived from a spreadsheet for the Poisson function. The table shows that $m = pN$ should be between 10 and 100. To make $m = 10$ (100) for a probability value of 2^{-13} would require a sample size of 81,920 (819,200).

Applying the equation for excess bit rate to a sample size of 81,920, and assuming that symbols having probabilities less than 2^{-13} contribute negligibly to the excess bit rate, gives an estimated excess bit rate of about 0.002 bit per symbol. Thus, a code derived from a sample in the order of 10^5 to 10^6 symbols should very closely approximate that derived from the entire pool.

Huffman coding is more tolerant of small probability estimation errors than information coding, because Huffman code word lengths are always integers. Therefore, very small estimation errors may not affect the Huffman code at all.

Table B.1

Probability that dp is within 10 (20) percent of p as a function of m = pN

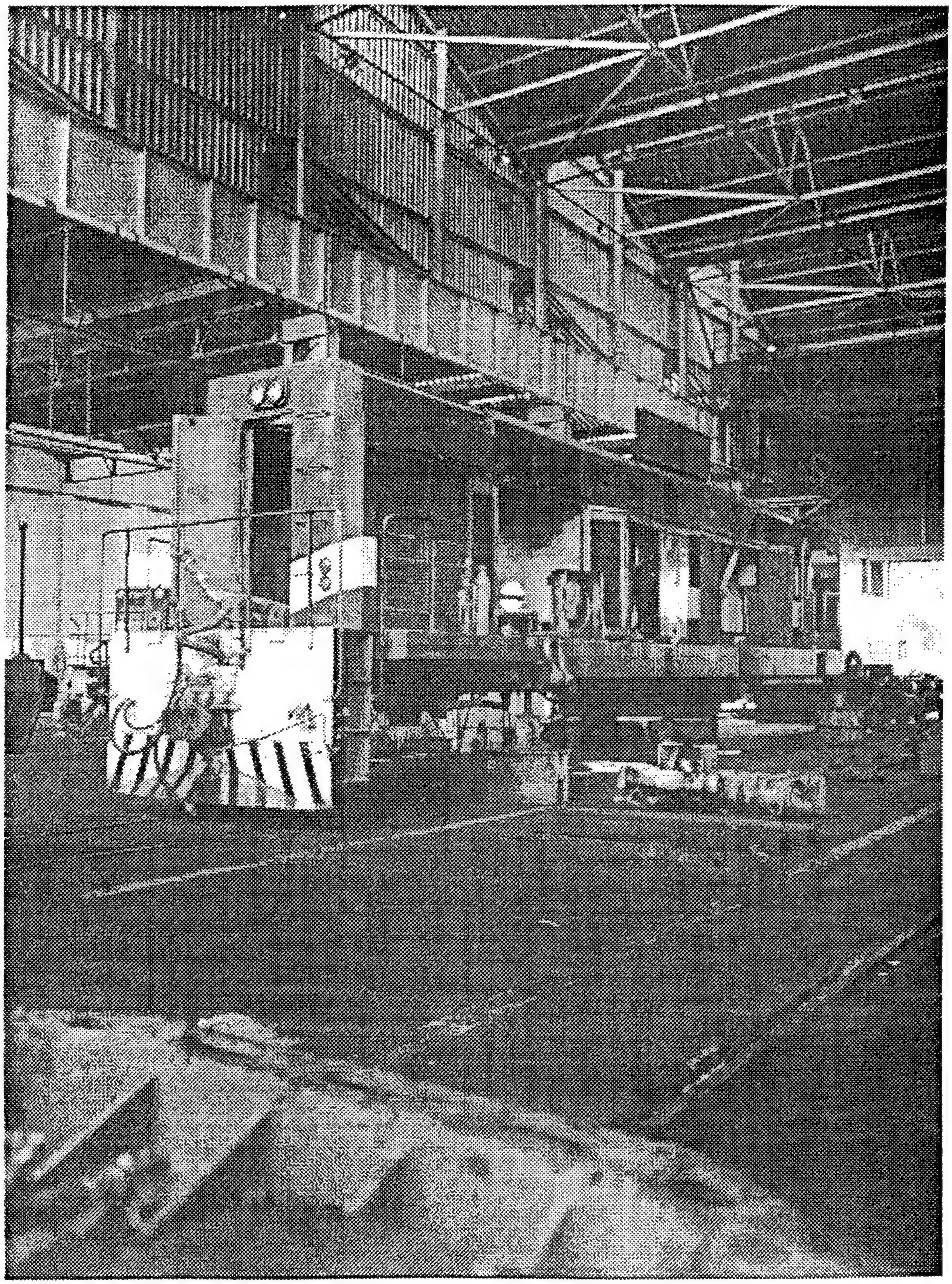
m	Pr. (10 percent)	Pr. (20 percent)
10	0.239	0.459
20	0.339	0.622
40	0.469	0.792
60	0.559	0.878
80	0.627	0.926
100	0.682	0.955

APPENDIX C

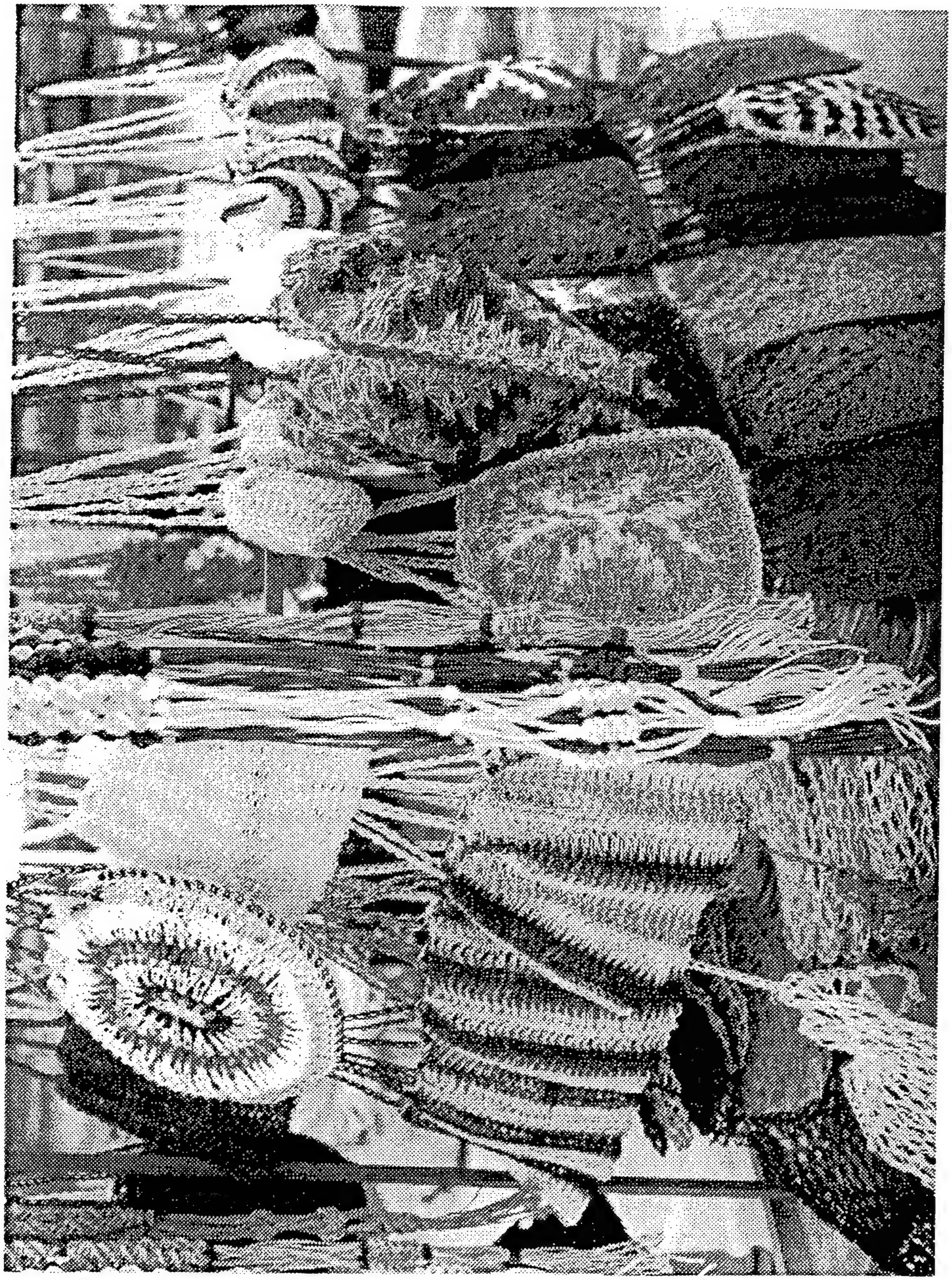
BUSY AND BLAND IMAGES FOR GENERATION OF DEFAULT HUFFMAN CODES

IMG0009.DLB

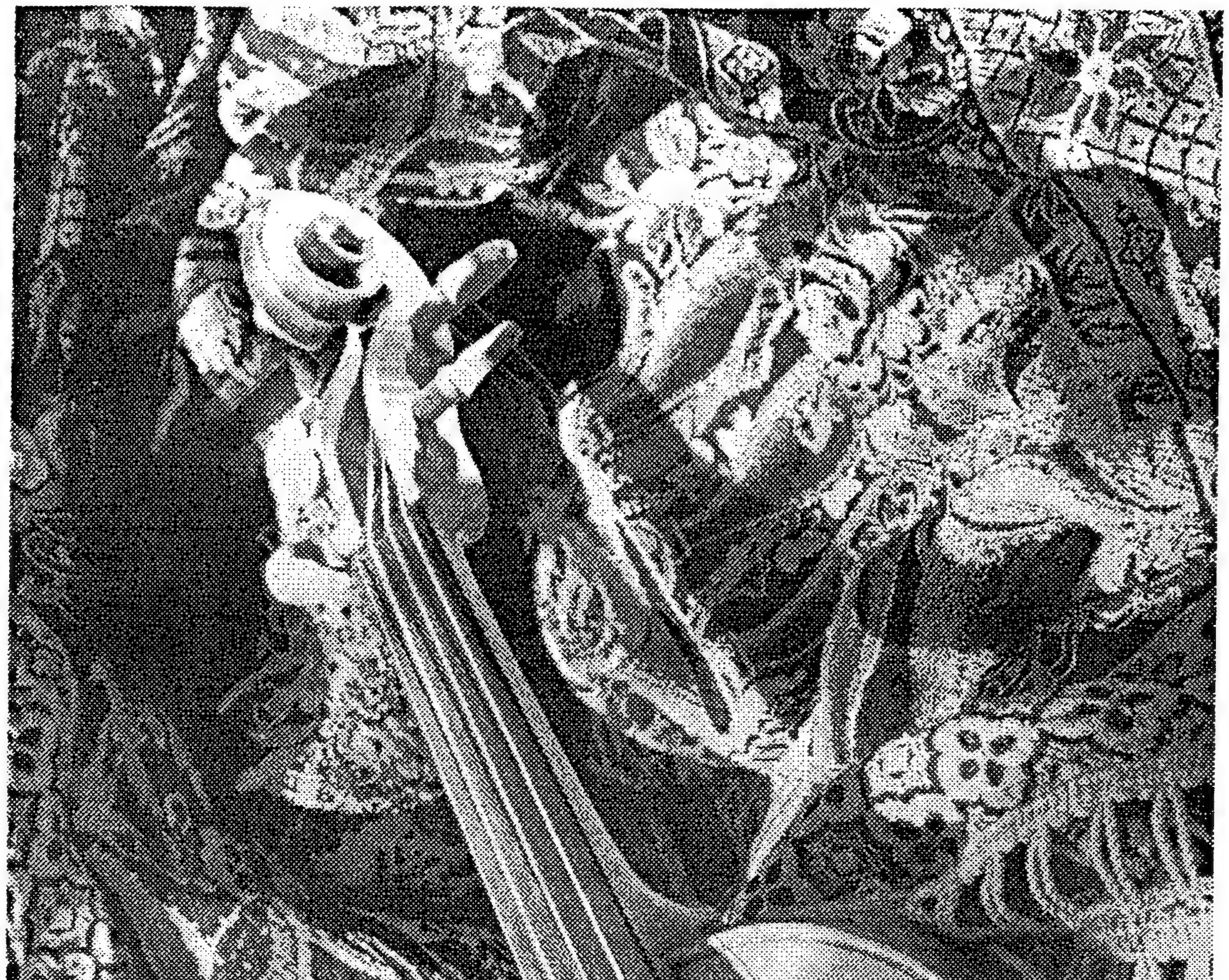
IMG0005.DLB



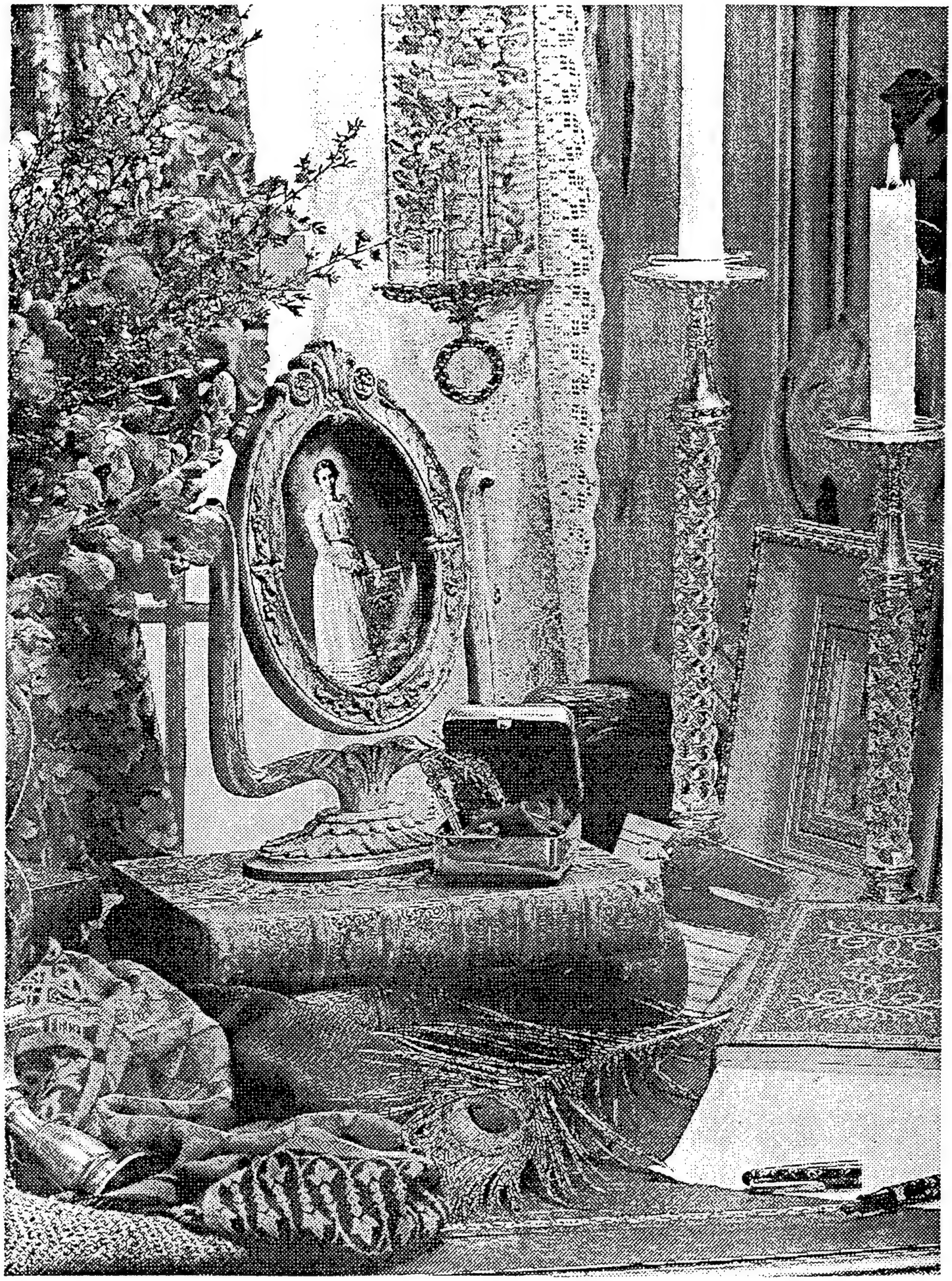
IMG0024.DLB



IMG0012.DLB

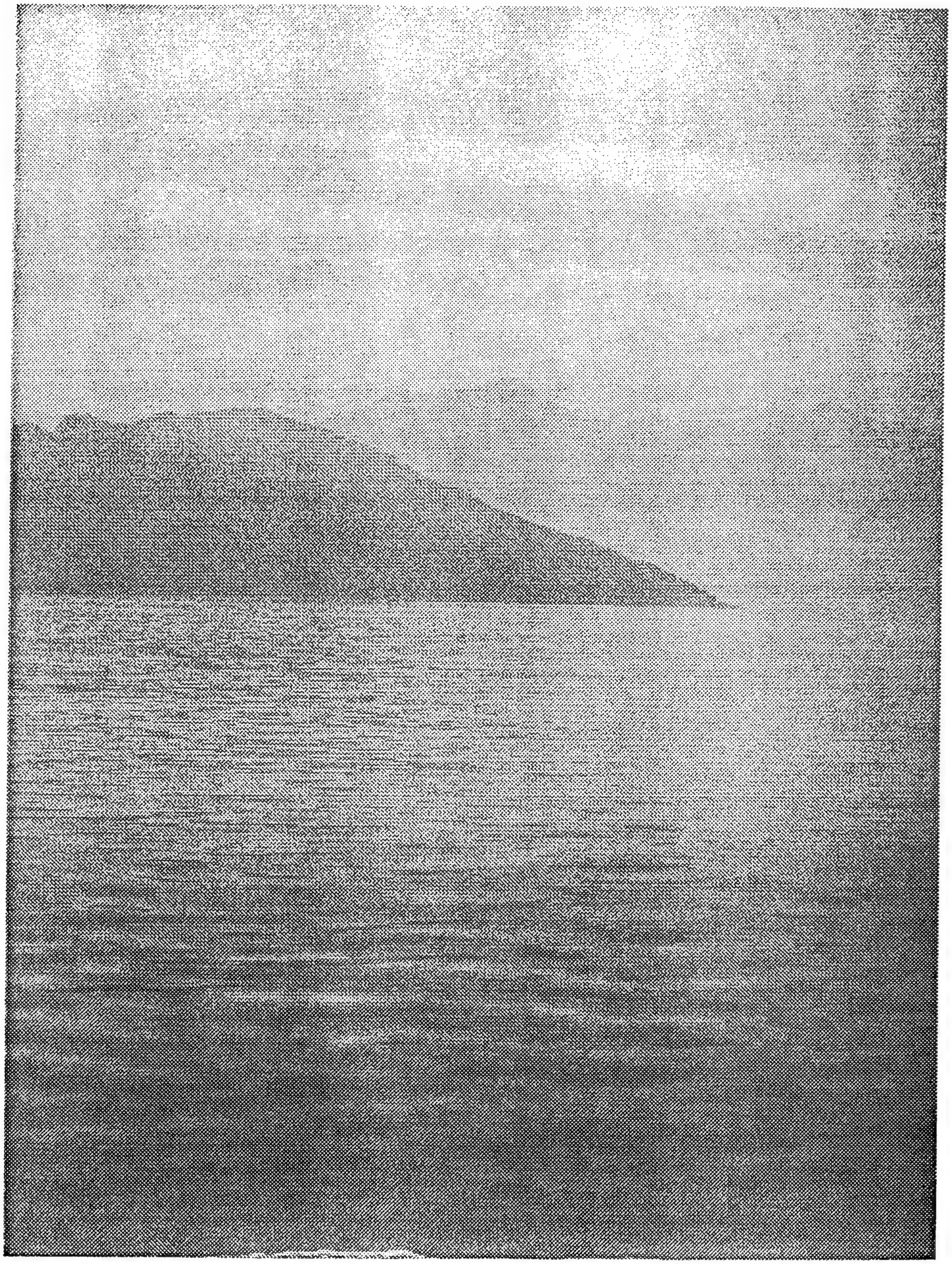


IMG0021.DLB

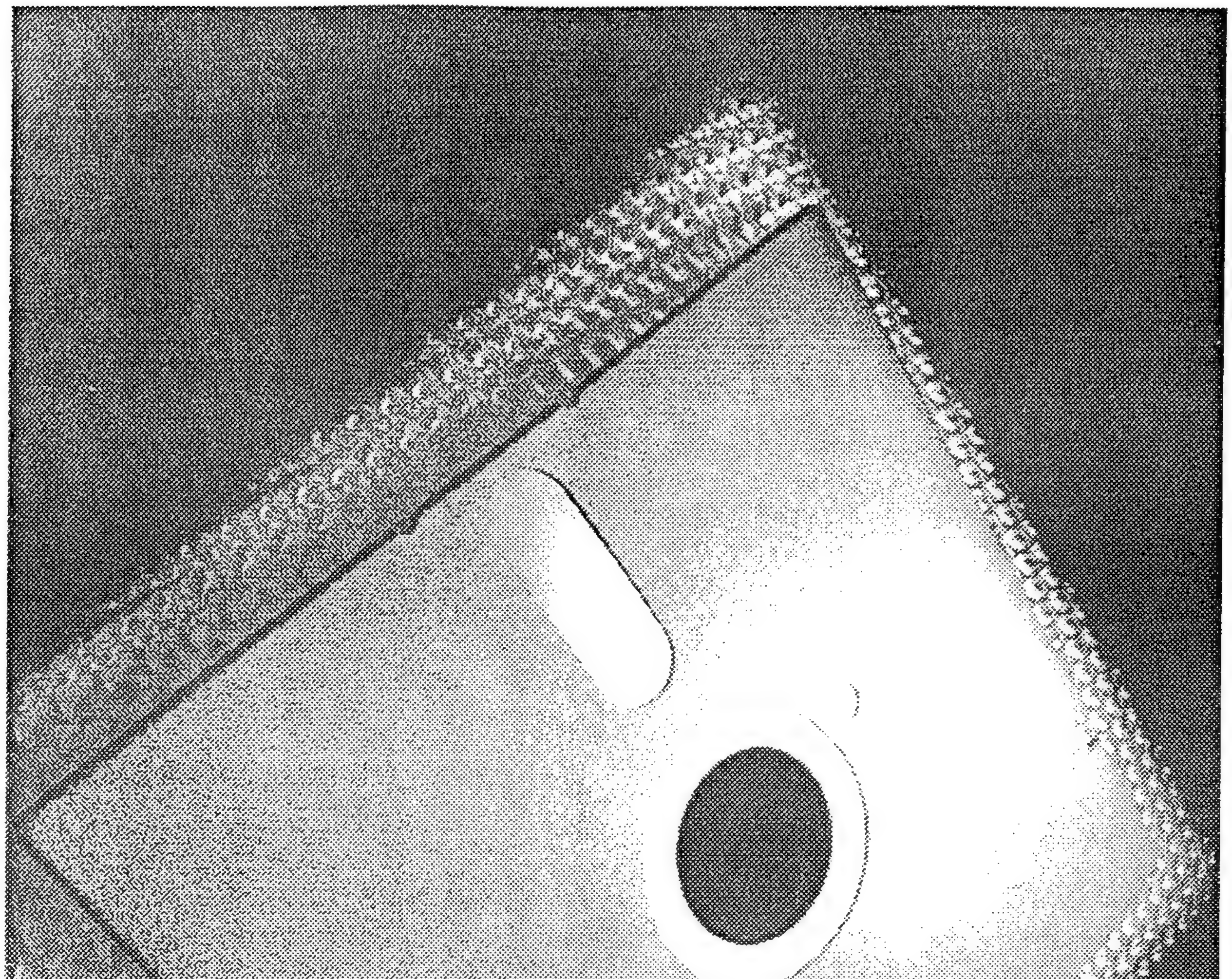


N8.DLB

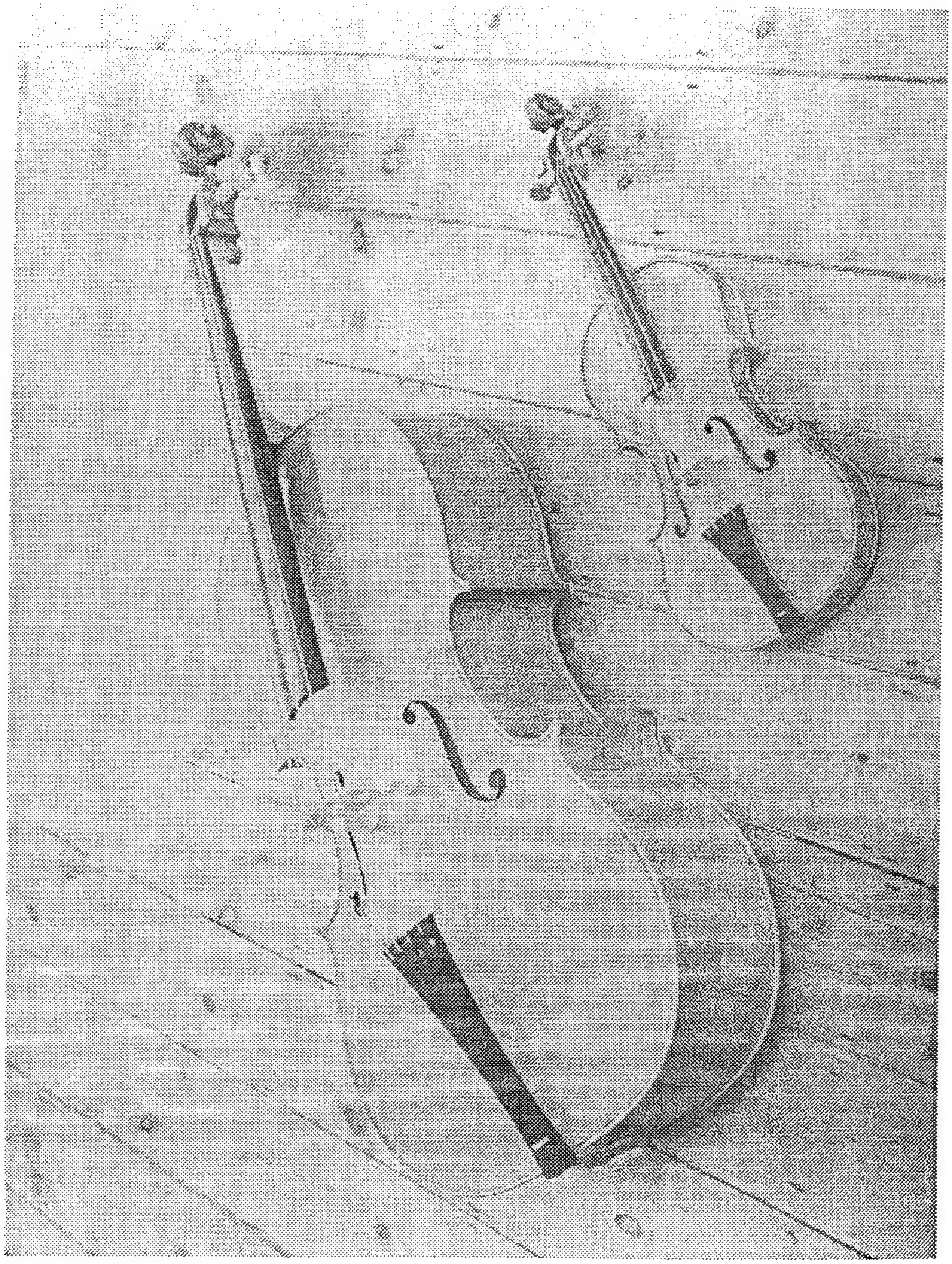
BIKERACE.DLB



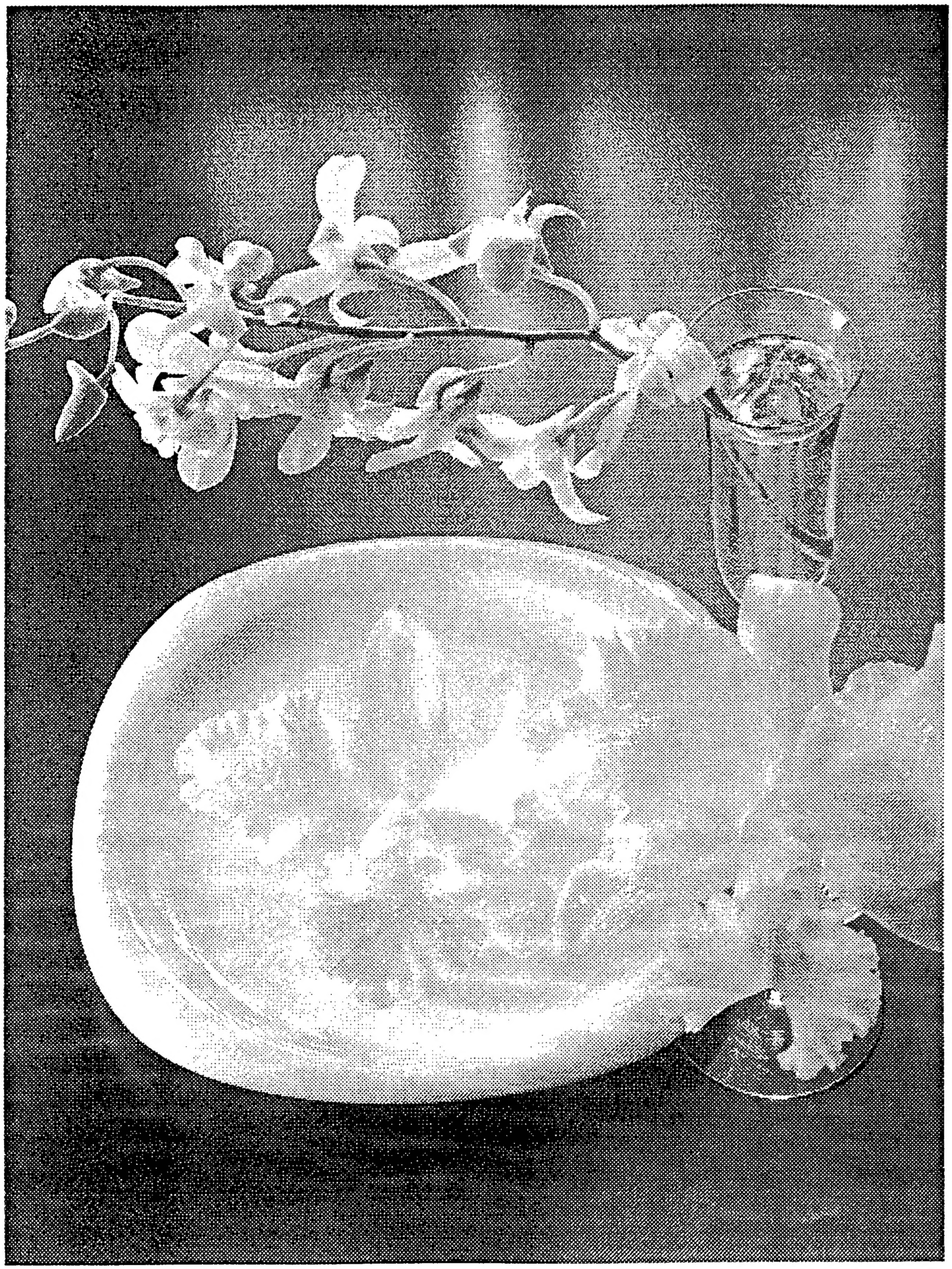
IMG0008.DLB



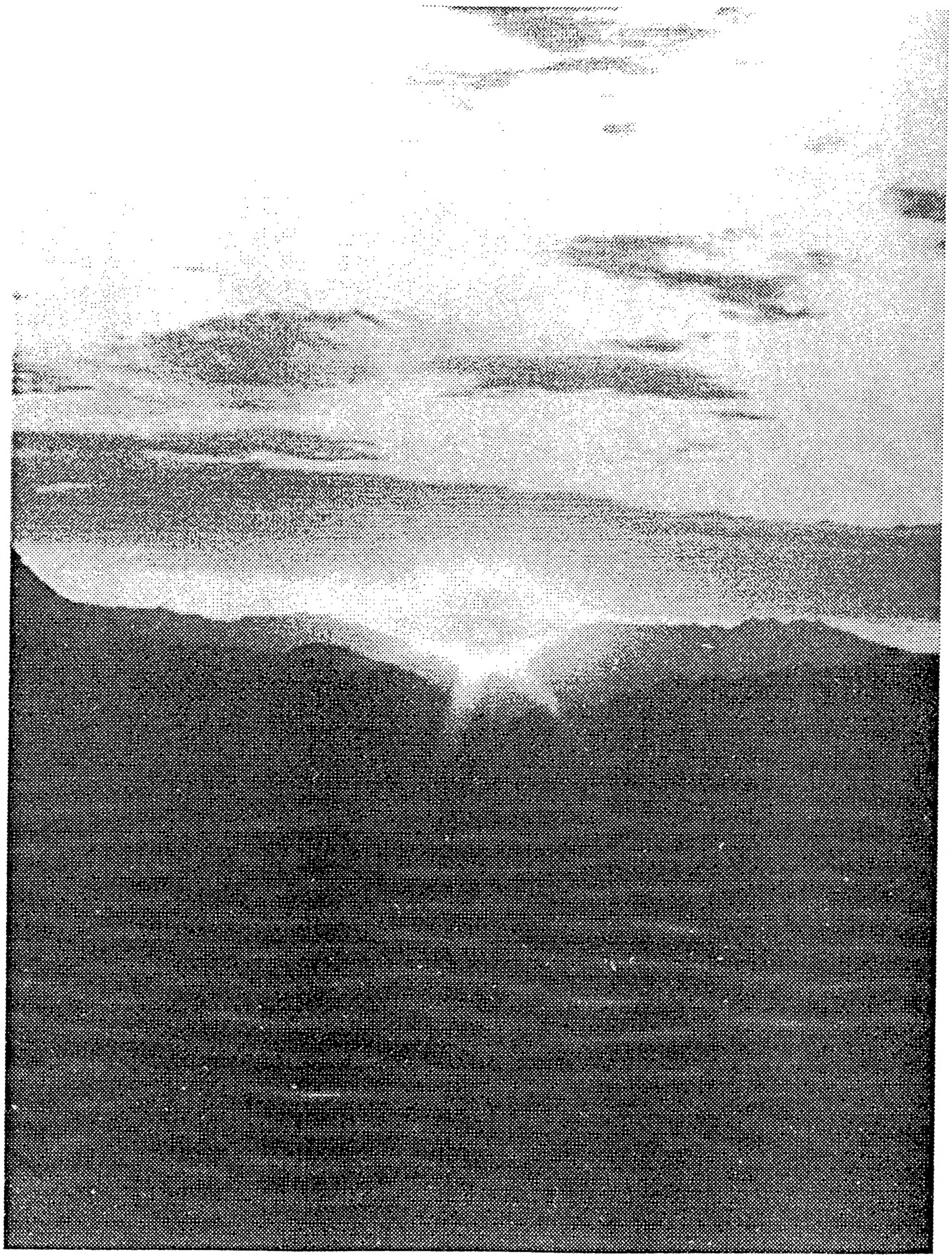
IMG0003.DLB



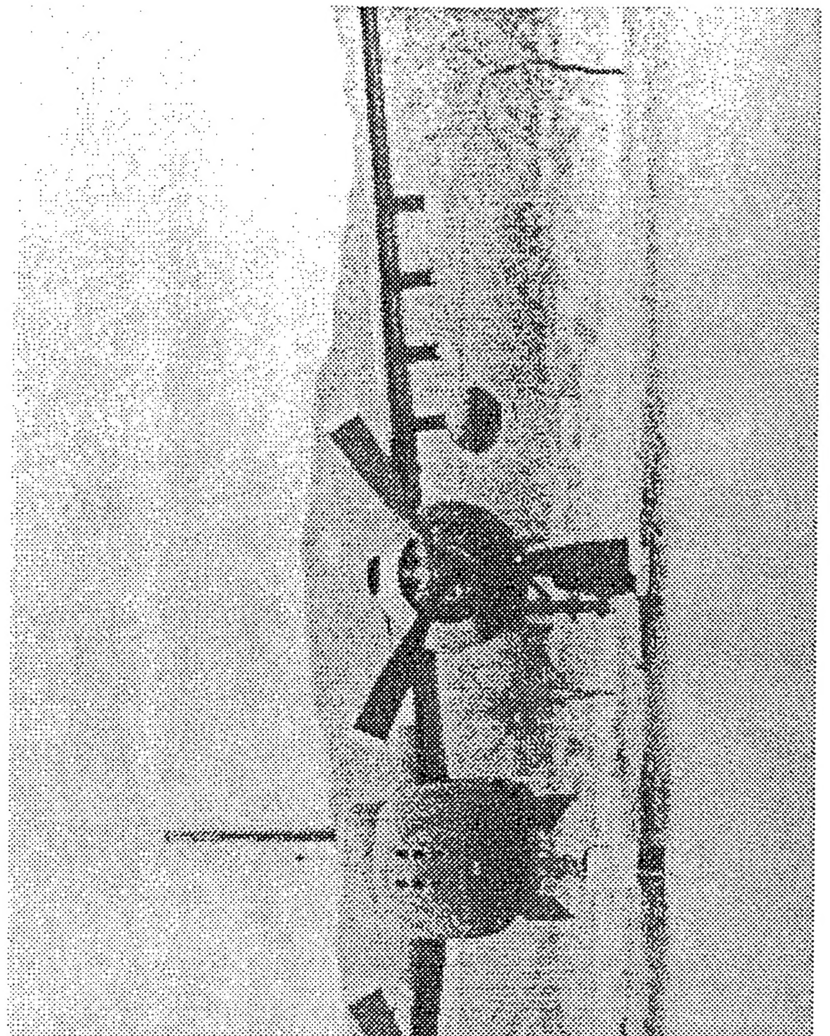
IMG0020.DLB



N6.DLB



SUNSET.DLB



PLANE.DLB

GCANYON.DLB